INTEGRATING INDEPENDENT AND CENTRALIZED MULTI-AGENT REINFORCEMENT LEARNING FOR TRAFFIC SIGNAL NETWORK OPTIMIZATION

A Thesis
Presented to
The Academic Faculty

By

Zhi Zhang

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Computer Science

Georgia Institute of Technology

May 2019

INTEGRATING INDEPENDENT AND CENTRALIZED MULTI-AGENT REINFORCEMENT LEARNING FOR TRAFFIC SIGNAL NETWORK OPTIMIZATION

Approved by:

Dr. Hongyuan Zha, Advisor
School of CSE
Georgia Institute of Technology

Dr. Tobin Isaac
School of CSE
Georgia Institute of Technology

Dr. Xiaojing Ye
School of Math
Georgia State University

Date Approved: April 25, 2019

To my parents and Will

ACKNOWLEDGEMENTS

First and foremost I offer my sincerest gratitude to my supervisor, Prof. Hongyuan Zha, who has supported me throughout my thesis with his patience and knowledge. I attribute the level of my Masters degree to his guidance, encouragement and effort and without him this thesis, too, would not have been completed or written. I also thank the members of my thesis committee: Dr. Tobin G. Isaac and Dr. Xiaojing Ye for their guidance and suggestions. The smooth running of my project is much more a team effort than my own. I am also greatly grateful for the help of my lab mate Jiachen Yang, who co-worked with me to achieve much progress. He showed me how to work independently to complete the research work, I learned a lot from him. I also want to thank my friends Rakshit Trivedi and Yujia Xie, who gave me encouragement and help during not only in study, but also in daily life. Finally, I wish to thank my parents in China, for the sacrifices they have made for me, for their support on my decision to receive a better education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Traffic congestion in metropolitan areas is a world-wide problem that can be ameliorated by traffic lights that respond dynamically to real-time conditions. Recent studies applying deep reinforcement learning (RL) to optimize single traffic lights have shown significant improvement over conventional control. However, optimization of global traffic condition over a large road network fundamentally is a cooperative multi-agent control problem, for which single-agent RL is not suitable due to environment non-stationarity and infeasibility of optimizing over an exponential joint-action space. Motivated by these challenges, we propose QCOMBO, a simple yet effective multi-agent reinforcement learning (MARL) algorithm that combines the advantages of independent and centralized learning. We ensure scalability by selecting actions from individually optimized utility functions, which are shaped to maximize global performance via a novel consistency regularization loss between individual utility and a global action-value function. Experiments on diverse road topologies and traffic flow conditions in the SUMO traffic simulator show competitive performance of QCOMBO versus recent state-of-the-art MARL algorithms. We further show that policies trained on small sub-networks can effectively generalize to larger networks under different traffic flow conditions, providing empirical evidence for the suitability of MARL for intelligent traffic control.

CHAPTER 1
INTRODUCTION AND BACKGROUND

With increasing urbanization, traffic congestion is becoming a significant and costly problem [1, 2]. While many early works proposed to optimize traffic light controllers based on expert knowledge and traditional model-based planning [3, 4, 5], there is increasing recent interest in applying flexible model-free methods in reinforcement learning (RL) [6] and deep RL, such as DQN in particular [7], to find optimal policies for traffic light controllers that dynamically respond to real-time traffic conditions [8, 9, 10, 11]. These works model a single traffic light as a Markov decision process (MDP) equipped with a small discrete action space (e.g. signal phase change) and a high-dimensional continuous state space (e.g. vehicle waiting time, queue length, etc.), and train a policy to optimize the expected return of an expert-designed reward function.

However, the single-agent RL perspective on traffic control optimization fails to account for the fundamental issue that optimizing global traffic flow over a densely interconnected road network is a multi-agent cooperation problem, where independently-learning agents face difficulty finding global optimal solutions. For example, a traffic intersection with low vehicle density in the North-South direction should let East-West traffic flow with little interruption to maximize its own performance, but this greedy policy will cause high East-West traffic load for an adjacent intersection that must also accommodate heavy traffic in the North-South direction. Instead, each traffic light agent must learn to act cooperatively to maximize global traffic network conditions, even while it optimizes its own individual reward based on observations of local conditions. Achieving high global performance without excessive negative impact to any single intersection requires a learning algorithm to account for actions that could produce complex interactions that propagate across the whole network.

On the other hand, existing work that adopt the multi-agent perspective on traffic signal optimization either fall back to independent learning [12] or resort to centralized optimization of coordinated agents [13, 14], both of which have disadvantages. While independent learning [15] can be naïvely applied to such decentralized cooperative multi-agent problems where each agent only optimizes its own reward based on local observations, independent learners cannot optimize for global criteria, such as different priorities for different intersections, and they face a nonstationary environment due to other learning agents, which violates the Markovian assumptions of RL algorithms. While centralized training can leverage global information to find cooperative optima, it requires maximization over a combinatorially-large joint action space and faces scalability issues both in training and in deployment.

Motivated by these challenges, my method focuses on deep multi-agent reinforcement learning (MARL) for traffic signal control with the following specific contributions:

1. Novel objective function combining independent and centralized training. We propose QCOMBO, a Q-learning based method with a new objective function that combines the benefits of both independent and centralized learning (Figure 3.1). The key insight is to learn a global action-value function based on global reward information to shape the learning of independently learning agents, who may otherwise settle at non-cooperative local optima. Specifically, we employ agent-specific observations and rewards for fast independent learning of local utility functions, learn a global action-value function using the global reward under the joint policy, and enforce consistency between local and global action value functions via a novel regularizer in a single objective function.

2. Evaluation of state-of-the-art MARL algorithms on traffic signal optimization. Recent cooperative MARL algorithms work in the paradigm of centralized learning with decentralized execution, specifically for the case when all agents share a single global reward [16, 17, 18]. However, as these methods were not designed for settings with individual rewards, it is open as to whether their performance can be surpassed by leveraging

such agent-specific information. While these algorithms have shown promise on complex multi-agent games, to the best of our knowledge they have not been tested on the important real-world problem of optimizing traffic signal over a network. Hence we conducted extensive experiments comparing our algorithm versus independent Q-learning (IQL), independent actor-critic (IAC), COMA [16], VDN [17] and QMIX [18] on a variety of road networks with varying traffic conditions.

3. Generalizability of traffic light control policies. Reinforcement learning methods are especially suitable for dynamic traffic light control since the transfer of a policy learned in simulation to real-world execution is arguably more feasible than in other domains (e.g. robotics). Similar to domains where deep RL excels [7], each traffic light has a small set of discrete actions for a signal phase change, which poses negligible issues for sim-to-real transfer. Furthermore, given improvements in sensor technology, measurements of traffic conditions (e.g. vehicle waiting time) can be increasingly accurate and hence real-world measurements may approach ideal simulated data. Hence, there is strong motivation to investigate whether a decentralized policy trained with simulated traffic approximating real-world conditions can be transferred to larger real-world networks and different traffic conditions without loss of performance. To the best of our knowledge, we conduct the first investigation on the generalizability and transferability of deep MARL policies for traffic signal control.

CHAPTER 2

RELATED WORK

Early work demonstrated the application of RL to single traffic light control [8]. The success of deep RL has spurred numerous recent works on incorporating higher-dimensional state information into a more realistic problem definition [9, 19, 11, 20]. In this paper, we extended the definition of states and actions in these works to define the observation space and action space of our multi-agent setting. Various choices of the reward function were proposed for training a single traffic light agent, such as vehicle delay and queue length [21, 22, 19]. In our multi-agent setting, we extended the definition of a single-agent reward as a linear combination of multiple features [11], by defining the global reward as a weighted sum of individual rewards using the PageRank algorithm [23].

Previous work on multi-agent traffic light control mostly relied on independent Q-learning (IQL) with heuristics to account for nonstationarity and coordination, such as: single-agent Q-learning for a central intersection surrounded by non-learning agents [13]; applying a Q function learned on a sub-problem to the full problem with max-plus action-selection [14]; training only one agent during each episode while fixing other agents' policies [12]; sharing information among neighboring Q-learning agents in either a discrete-state [24] or continuous-state formulation [25]. However, these approaches do not account for the importance of macroscopic measures of traffic flow [26]; in contrast, our formulation explicitly shapes the learning of individual agents via a global reward.

To address the difficulties with independent learning methods [15] such as IQL and independent actor-critic (IAC), recent work have proposed more sophisticated deep MARL algorithms for cooperative multi-agent problems with a global reward that explicitly address the need for scalable centralized training with decentralized execution. Counterfactual multi-agent policy gradients (COMA) [16] demonstrated the benefit of cooperative

multi-agent credit assignment in StarCraft micromanagement; VDN [17] improved scalability of centralized training by decomposing the global action-value function into individual utilities; QMIX [18] generalized VDN by using a hypernetwork to enforces the monotonicity of a nonlinear mixing function mapping individual utilities to the global action-value function. However, these methods only learn from a global reward without using available individual rewards, which motivates our proposal for a simple yet effective way to combine individual and centralized training. Moreover, to the best of our knowledge, these algorithms have yet to be evaluated and compared in the real-world problem of multi-agent traffic light control, which motivates one of our main contributions.

CHAPTER 3

PROBLEM FORMULATION AND METHOD

3.1 Markov game model of multi-agent traffic light control

We formulate the multi-agent traffic light control problem as a partially-observed Markov game $\langle S, \{O\}^n, \{A\}^n, P, R, N, \gamma \rangle$, consisting of N agents labeled by $n = [1..N]$, defined as follows:

Agents $n \in [1..N]$. Each agent controls the phase of one traffic light at an intersection.

Observation space O^n. All agents share the same observation space $O := O^1 = \cdots = O^N$, since we assume they have the same measurement capabilities. Each $o^n \in O$ represents agent n's individual limited observation, with the following components: $q^n \in \mathbb{R}^l$, $v^n \in \mathbb{R}^l$, $wt^n \in \mathbb{R}^l$, $delay^n \in \mathbb{R}^l$, for l incoming lanes of a traffic light, and the $ph^n \in \mathbb{R}^2$, $d^n \in \mathbb{R}$:

- q^n: the length of queue on incoming lanes, defined as the total number of halting vehicles (speed less than 0.1m/s);
- v^n, the number of vehicles on each incoming lane;
- wt^n, the average waiting time of all vehicles on each incoming lane; defined as the time in minutes a vehicle spent with a speed below 0.1m/s since the last time it was faster than 0.1m/s
- $delay^n$, the average delay of all vehicles on each incoming lane, the delay of a lane is equal to $1 - \frac{\text{average vehicle speed}}{\text{maximum allowed vehicle speed}}$;
- ph^n: the traffic light's current phase, indicating the status of the east-west and north-south directions, represented by a 0-1 one-hot variable such that $ph^n : EW \times NS \mapsto \{0, 1\}^2$;

- d^n: current phase duration in seconds since the last phase change.

Prior work used an image representation of positions of all vehicles near a traffic light as part of the observation, processed using convolutional networks Wei, Zheng, Yao, and Li [11]. In contrast, we show this is not necessary and hence significantly reduce computational cost.

Global state space S contains all the global information on the current system, including every route, and traffic light, and S is equal to the collection of all individual observations.

Action space A^n. All agents share the same action space $A := A^1 = \cdots = A^N$, as we assume that all traffic signal controllers have the same capabilities. The action a^n of each agent is a binary variable to indicate whether or not the traffic light will keep the current phase or switch to another phase. The game has joint action space $\mathbf{A} \equiv A^1 \times A^2 ... \times A^n$, and each time step the system has joint action $\mathbf{a} := (a^1, \ldots, a^N) \in \mathbf{A}$. We use $\mathbf{a}^{-n}$ to denote the collection of all actions *except* that of agent n.

Individual reward $R(s, \mathbf{a}) : S \times \mathbf{A} \mapsto \mathbb{R}$. Since previous work have shown the feasibility of using weighted route features as the reward for the single-agent traffic light setting [14, 11], we directly applied that to obtain the individual reward. There are seven features used to calculate the individual reward, described in Section A.2. The individual reward is a combination of these meaningful features that capture many intuitive metrics of desirable and undesirable traffic conditions.

Global reward R^g, defined as a weighted sum of individual rewards. We explored different methods to compute these weights. We used the PageRank algorithm to compute weights on each individual reward [23], since traffic intersections with higher risk of congestion are generally located in the central areas of the map due to higher interactions with surrounding traffic, and therefore should receive higher priority. Hence $R^g(s, \mathbf{a}) := \sum_n k_n R^n$, where $k_n = \text{PageRank}(n)$. Alternatively, we considered using the traffic flow conditions under a fixed control policy to compute the weights for each traffic lights; how-

ever, this formulation is not a good choice since it is not clear which the policy should be used to set the "nominal" traffic condition, and a suboptimal policy can trigger a bad estimation of weights. In contrast, the PageRank algorithm accounts for the topological structure of the transportation network, addresses the connectivity and interaction between agents, and assigns more weights to the reward of a traffic light that is highly connected with others. Thus the whole global reward considers both the static structure and dynamic flow of the operating traffic.

Evaluation criteria. Given a reward function designed with sufficient expert domain knowledge and specified with enough precision to disambiguate different traffic states, we can investigate the performance of state-of-the-art MARL algorithms by directly evaluating them using the cumulative reward and components of the reward. Hence we do not resort to manual inspection of policy behavior, in contrast to previous work where certain states were aliased (i.e. produce the same reward) and manual inspection of the policy was required Wei, Zheng, Yao, and Li [11].

3.2 Multi-agent Reinforcement Learning Algorithms

3.2.1 IQL and IAC

Independent Q-learning (IQL) and independent actor-critic (IAC) have demonstrated surprisingly strong performance in complex multi-agent systems [15, 27, 28, 18]. IQL directly applies single-agent Q-learning to each agent of the Markov game. While the optimal action-value function of an MDP is defined as

$$Q^*(s,a) := \max_{\pi} \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t R_t \mid s_0 = s, a_0 = a\Big], \tag{3.1}$$

IQL agents learn a local utility function $Q(o^n, a^n)$ by minimizing the loss function [7]

$$L(\theta) = \mathbb{E}_{\pi}\Big[\big(r_t + \gamma \max_{a} Q(s_{t+1}, a; \theta') - Q(s_t, a_t; \theta)\big)^2\Big] \tag{3.2}$$

where θ are parameters of the function approximation and θ' are parameters of a target network. For each agent n, IQL learns a local utility function $Q^n(o^n, a^n)$, which is not a true action-value function because the presence of other learning agents result in a nonstationary environment from any single agent's perspective. Similarly, IAC directly applies the single-agent policy gradient with a variance-reduction baseline $b(s)$ [29]

$$\nabla_\theta J(\pi) = \mathbb{E}_\pi\Big[\nabla_\theta \log \pi(a|s)\big(Q^\pi(s,a) - b(s)\big)\Big] \tag{3.3}$$

to train an actor-critic pair for each agent, resulting in actors $\pi^n(a^n|o^n)$ and critics $Q(o^n, a^n)$. While IQL and IAC agents display strong ability to optimize individual rewards [15, 30], the lack of global information and a mechanism for cooperation means they are likely to settle for sub-optimal solutions.

3.2.2 COMA

In cooperative MARL with a single global reward, COMA [28] estimates a centralized action-value function $Q^{\boldsymbol{\pi}}(s, \mathbf{a})$ and uses it in a counterfactual baseline for reducing variance of a multi-agent policy gradient:

$$\nabla_\theta J(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_n \nabla_\theta \log \pi^n(a^n|o^n)\big(Q^{\boldsymbol{\pi}}(s,\mathbf{a}) - b(s,a^{-n})\big)\Big] \tag{3.4}$$

$$b(s,a^{-n}) := \sum_{\hat{a}^n} \pi^n(\hat{a}^n|o^n) Q^{\boldsymbol{\pi}}(s,(a^{-n},\hat{a}^n)) \tag{3.5}$$

The counterfactual baseline $b(s, a^{-n})$ improves multi-agent credit assignment, since the advantage $A^n(s,\mathbf{a}) := Q^{\boldsymbol{\pi}}(s,\mathbf{a}) - b(s,a^{-n})$ evaluates the contribution of an agent's chosen action a^n versus the average of all possible counterfactuals $\hat{a}^n$, keeping a^{-n} fixed. Their formulation is a low variance gradient estimate, as the advantage function evaluates the contribution of an agent's chosen action a^n versus the average of all possible counterfactuals $\hat{a}^n$, keeping other agents' a^{-n} fixed. However, since the only learning signal comes

from the global reward and individual agents are not directly trained to improve local performance, COMA may exhibit slower training in cooperative traffic light control.

3.2.3 VDN

While IQL agents cannot learn to cooperate for a global reward, it is also not feasible to learn a single optimal action-value function $Q^*(s, \mathbf{a})$ since the maximization step requires searching over $|\mathcal{A}|^N$ joint actions. Instead, VDN [17] proposed to learn a joint action-value function $Q^{\text{VDN}}(s, \mathbf{a})$ that decomposes as $Q^{\text{VDN}}(s, \mathbf{a}) := \sum_{n=1}^{N} Q_n(o^n, a^n)$, so that $Q^{\text{VDN}}(s, \mathbf{a})$ is trained with

$$L(\theta) = \mathbb{E}_{\boldsymbol{\pi}}\Big[\big(y_t - Q^{\text{VDN}}(s_t, \mathbf{a}_t)\big)^2\Big] \tag{3.6}$$

$$y_t := R + \gamma Q^{\text{VDN}}(s_{t+1}, \mathbf{a}_{t+1})|_{\mathbf{a}_{t+1}=\{\text{argmax}_{a^n}\, Q(o^n_{t+1}, a^n)\}_n} \tag{3.7}$$

Agents act greedily with respect to their own utility functions $Q(o^n, a^n)$, while global reward is used for overall training. However, there is no guarantee in general settings that the true optimal $Q^*(s, \mathbf{a})$ can be decomposed as a linear combination of individually-optimized utilities, which could limit VDN's performance.

3.2.4 QMIX

QMIX [18] generalizes VDN by representing the optimal action-value function as a nonlinear function $Q^*(s, \mathbf{a}) = F(Q_1, \ldots, Q_N)$ of individual utility functions, while ensuring that the combination of individual argmax on each Q_n yields the same joint action as a global argmax on $Q^*(s, \mathbf{a})$. This is achieved by enforcing positive weights in the nonlinear mixing network F. QMIX enforces the monotonicy of the function by using hypernetworks, which passes the state variable through linear layers followed by absolute-activation functions, to generate the weights of a mixing network that takes in all agents' Q_n and produces a joint Q value. Despite being a more expressive model than VDN, the stability of QMIX depends

on appropriate choices of mixing network architecture, for which there is little theoretical guidance, and QMIX also relies on global reward without using local reward for training.

3.3 Method

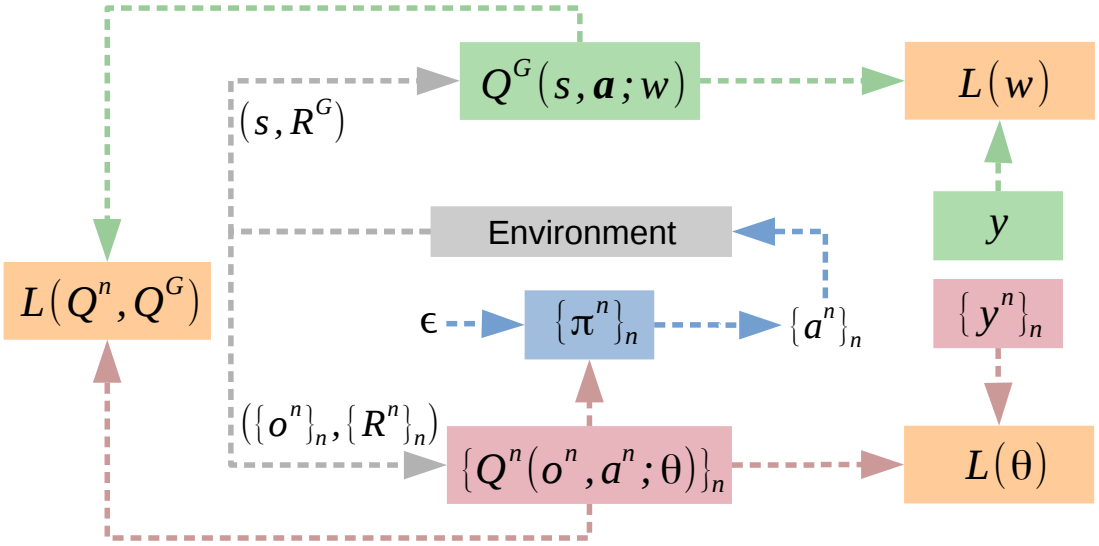


Figure 3.1: QCOMBO architecture combining independent learning of $Q^n(o^n, a^n)$ with centralized training of $Q(s, \mathbf{a})$ via a novel consistency loss $L(Q, \{Q^n\}_n)$

We propose QCOMBO, a combination of value decomposition network and independent Q learning with a new consistency regularizer. Optimal policies for each agent are learned by optimizing a mixed objective consisting of three parts: an individual term based on the loss function of independent DQN, a global term for learning a global action-value function, and a shaping term that minimizes the difference between the weighted sum of individual Q values and the global Q value. This algorithm ensures that agents learn to maximize the global reward, which is difficult for independently-learning agents to achieve, and also maintain the ability to optimize their individual performance using agent-specific observations and rewards, which is difficult under a purely centralized approach. This section describes the three components in sequence.

3.3.1 Individual Part

Using individual observations and rewards for each agent is computationally efficient, since local observations generally have lower dimension than global state information, and also algorithmically efficient, since it avoids the difficult credit assignment problem of decomposing a single global reward signal for each agent to learn. Furthermore, by optimizing individual utility functions Q^n instead of a global optimal Q function, we reduce the maximization problem at each step of Q-learning from $O(|\mathcal{A}|^N)$ to $O(N|\mathcal{A}|)$. From an individual perspective, each agent learns a policy conditioned only on its own observations to maximize its own expected cumulative reward. Parameterizing the local utility function for agent n with parameter θ^n, we minimize the loss

$$\mathcal{L}(\theta^n) = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\boldsymbol{\pi}}\Big[\frac{1}{2}(y_t^n - Q^n(o_t^n, a_t^n; \theta^n))^2\Big] \tag{3.8}$$

$$y_t^n = r_t^n + \gamma \max_{\hat{a}^n} Q^n(o_t^n, \hat{a}^n, \theta'^n) \tag{3.9}$$

Since the agent population is homogeneous (i.e. all agents have the same observation and action spaces), we improve memory and computational efficiency by employing *parameter-sharing* [16] among all agents, which means $\theta := \theta^n, \forall n \in [1..N]$. Agents still act differently since they receive different observations, and we further give an agent indicator as input for agent disambiguation.

3.3.2 Global Part

Without global information, independently learning agents face a nonstationary environment due to the presence of other learning agents, and they may have insufficient information to find cooperative optima. On the other hand, training an optimal global Q function is not scalable, since the Q-learning step would require maximization over $|\mathcal{A}|^N$ possible joint actions for N agents. To address this dilemma, our key insight is that we can learn

the global Q function under the joint policy induced by all agents' local utility functions, rather than learn the optimal global Q function, and use it to shape the learning of individual agents via information in global state s and global reward R. Specifically, the joint policy defined by $\mathbf{a} \sim \boldsymbol{\pi}(\mathbf{a}|s) = \{\text{argmax}_{a^n} Q^n(o^n, a^n)\}_{n=1}^N$ is associated with a global action-value function

$$Q^{\boldsymbol{\pi}}(s, \mathbf{a}) := \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty} \gamma^t \sum_{n=1}^{N} k^n R_t^n(s, \mathbf{a}) \mid s_0 = s, \mathbf{a}_0 = \mathbf{a}\Big] \tag{3.10}$$

Parameterizing $Q^{\boldsymbol{\pi}}(s, \mathbf{a}; w)$ with w, we minimize the loss:

$$\mathcal{L}(w) = \mathbb{E}_{\boldsymbol{\pi}}\Big[\frac{1}{2}\big(y_t - Q^{\boldsymbol{\pi}}(s_t, \mathbf{a}_t; w)\big)^2\Big] \tag{3.11}$$

$$y_t = R_t + \gamma Q^{\boldsymbol{\pi}}(s_{t+1}, \mathbf{a}_{t+1}; w')|_{a_{t+1}^n = \text{argmax}_{a^n} Q^n(o_{t+1}^n, a^n; \theta')} \tag{3.12}$$

Crucially, note that action selection for computing the TD target (3.12) uses the greedy action from local utility functions (i.e. the induced policy) and does not need to consult the global Q function. Effectively, the collection of local utility functions induce a joint policy $\boldsymbol{\pi}$, which generate the data for off-policy learning of the global action-value function $Q^{\boldsymbol{\pi}}$.

3.3.3 Combined objective

If each agent greedily optimizes its own local utility function, the global return can be suboptimal. For example, if agent n (with low weight k^n) has no flow in the N-S direction while adjacent agent m (with high weight k^m) has heavy flow in the N-S direction, the individual optimal policy for n is to let W-E traffic flow continuously to m, which negatively impacts conditions at m and leads to low global reward. To address the suboptimality of independent learning, we propose a new consistency regularization loss

$$\mathcal{L}(Q, \{Q^n\}_n) := \mathbb{E}_{\boldsymbol{\pi}}\Big[\frac{1}{2}\big(Q(s, \mathbf{a}; w) - \sum_{n=1}^{N} k^n Q^n(o^n, a^n; \theta)\big)^2\Big] \tag{3.13}$$

between global Q and individual utility functions $\{Q^n\}_n$. Since Q is the true global action-value function with respect to the induced joint policy, this regularization brings the the weighted sum of individual utility functions closer to global expected return, so that the optimization of individual utility functions is influenced by the global objective rather than purely determined by local information. Hence the regularizer mitigates the potential issue that any individual agent attains high individual performance at the cost of hurting collective performance.

The complete QCOMBO architecture, shown in Figure 3.1, combines the individual loss (3.8), global loss (3.11), and consistency regularizer (3.13), and is trained by minimizing the overall objective:

$$\mathcal{L}(w, \theta) = \mathcal{L}(w) + \mathcal{L}(\theta) + \lambda \mathcal{L}(Q, \{Q^n\}_n) \tag{3.14}$$

where λ is a hyper-parameter that controls the extent of regularization. Whenever any agent learns to attains high individual reward at the cost of hurting global performance, which is likely to occur due to minimizing the individual loss, the consistency loss will increase to reflect the inconsistency between individual and global performance; it will then decrease as global information influences the learning of individual Q^n. Our experiments provide evidence of this dynamic learning process that balances individual and global learning (Figure B.1(a)).

At the i-th training iteration, we interleave the updates to each of θ_i and w_i, thereby achieving information exchange between the global and individual parts, allowing each agent to learn a policy that considers the effects of other agents' learning. The parameters

w_i and θ_i are updated by gradient descent with

$$\begin{aligned}
&\nabla_{w_i}\mathcal{L}(w_i,\theta_i) = -E_{\boldsymbol{\pi}}\Big[\big(y_t - Q^{\boldsymbol{\pi}}(s_t,\mathbf{a}_t,w')\big)\nabla_{w_i}Q^{\boldsymbol{\pi}}(s_t,\mathbf{a}_t,w_i) \\
&+\lambda\big(Q^{\boldsymbol{\pi}}(s_t,a_t,w_i) - \sum_n k^n Q^n(o^n,a^n,\theta_i)\big)\nabla_{w_i}Q^{\boldsymbol{\pi}}(s_t,a_t,w_i)\Big] \\
&= -E_{\boldsymbol{\pi}}\Big[\big[R^g + \gamma Q^{\boldsymbol{\pi}}(s_{t+1},\mathbf{a}_{t+1};w')|_{\mathbf{a}_{t+1}=\{\mathrm{argmax}_{a^n}\,Q(o^n_{t+1},a^n;\theta')\}_n} \\
&-(1-\lambda)Q^{\boldsymbol{\pi}}(s_t,\mathbf{a}_t,w_i) - \lambda\sum_n k^n Q^n(o^n_t,a^n_t,\theta_i)\big]\nabla_{w_i}Q^{\boldsymbol{\pi}}(s_t,\mathbf{a}_t,w_i)\Big]
\end{aligned} \tag{3.15}$$

$$\begin{aligned}
&\nabla_{\theta_i}\mathcal{L}(w_i,\theta_i) = -\,\mathbb{E}_{\boldsymbol{\pi}}\Big[\frac{1}{N}\sum_{n=1}^{N}\big(y^n_t - Q(o^n_t,a^n_t;\theta_i)\big)\nabla_\theta Q(o^n,a^n,\theta_i) \\
&+\lambda\Big(Q^{\boldsymbol{\pi}}(s_t,a_t;w_i) - \sum_{n=1}^{N}k^n Q(o^n_t,a^n_t;\theta_i)\Big)\nabla_{\theta_i}\sum_{n=1}^{N}k^n Q^n(o^n_t,a^n_t;\theta_i)\Big] \\
&= -\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{n=1}^{N}\Big(\big[R^n + \gamma\max_{\hat{a}^n}Q^n(o^n_{t+1},\hat{a}^n_t;\theta') - Q^n(o^n_t,a^n_t;\theta_i)\big] \\
&+\lambda k^n\big[Q^{\boldsymbol{\pi}}(s_t,\mathbf{a}_t;w_i) - \sum_{m=1}^{N}k^m Q^m(o^m_t,a^m_t;\theta_i)\big]\Big)\nabla_{\theta_i}Q^n(o^n_t,a^n_t;\theta_i)\Big]
\end{aligned} \tag{3.16}$$

In i iteration, the updating of θ and w depends on itself and the other from the $i-1$ iteration, this interactive iterative updating allows the information exchange from the global and individual part, allows each agent learns a policy with considering the effects of other agents' learning. Through a series of discrete time steps, the parameters are adjusted following the directions of the gradient of the loss, driving the approximator to select the strategies that maximize the cumulative discounted rewards. The learned individual policy will not only maximize the self reward, but also consider the whole system which contains other agents.

CHAPTER 4
EXPERIMENTS AND RESULTS

4.1 Experiment Setup

We evaluated the performance of all MARL algorithms described above on multiple road networks under a variety of traffic conditions in the SUMO simulator [31, 32]. We describe all key experimental setup details in this section, and devote Section 4.2 to detailed analysis of each algorithm's performance results. For each algorithm, we report the mean of five independent runs, with standard deviation reported in Table 4.2.

4.1.1 Environment

We configured our experimental setup using the Flow framework with the SUMO simulator. We used homogeneous vehicles of the same type and size in all experiments to control the variance of results, especially for policy gradient-based algorithms (e.g. IAC) that may exhibit high variance. Road networks are defined as the intersection of m horizontal and n vertical roads (e.g. a 1×2 network has one horizontal route intersecting two vertical routes.). Each traffic light situated at an intersection is a learning agent. Each road between two intersections is 400m long and has two lanes with opposite directions of travel. Hence each agent has one incoming and one outgoing lane for each edge. Vehicles are emitted at the global outer boundaries of each edge with random starting lane and fixed entering speed. We used different traffic flow programs which vary in the number of vehicles per hour in the specific period to show our findings. Table 4.1 contains all traffic configurations. For a given time step, the traffic light is green exclusively for either the horizontal or the vertical direction.

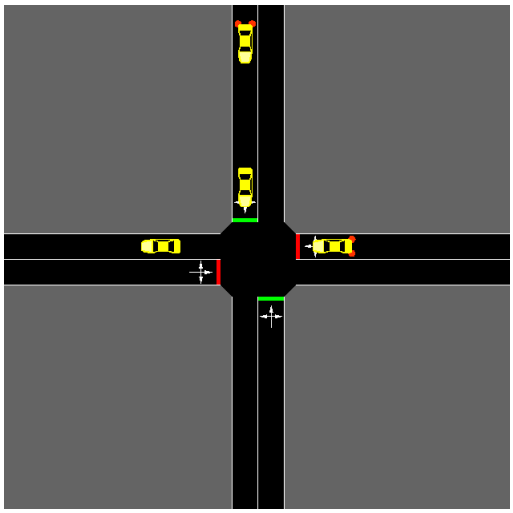


Figure 4.1: 1 traffic light example

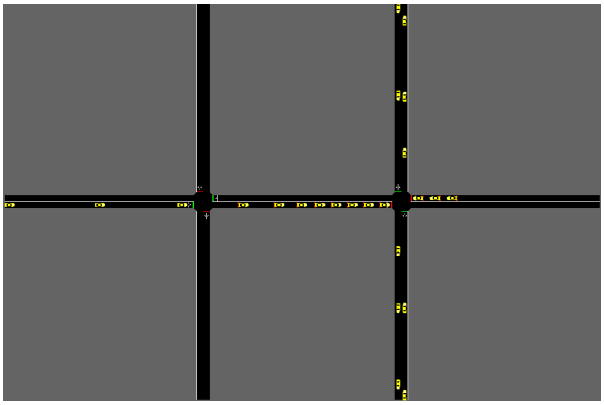


Figure 4.2: 2 traffic lights

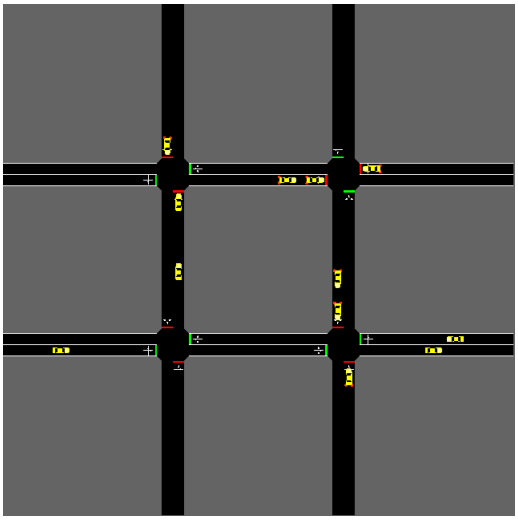


Figure 4.3: 2×2 network, 4 traffic lights

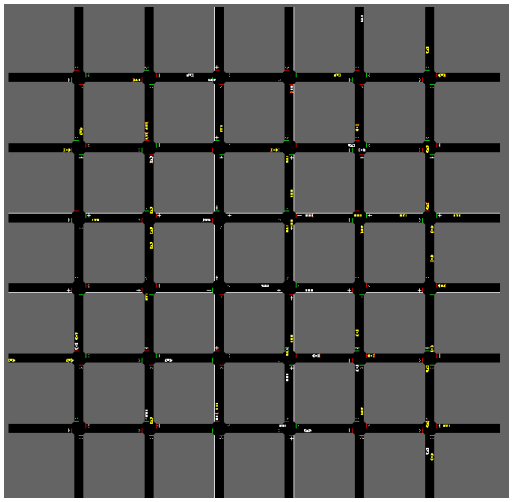


Figure 4.4: 6×6 traffic lights

Table 4.1: The Experiment Program Configuration Details

Program	Time Period	Train/Test	Topology	Horizontal (Bot to Top) num vehicles/hour	Vertical (Left to Right) num vehicles/hour
1×2 two Agents	0-12000	train	1×2	700	10, 620
2×2 Same State	0-12000	train	2×2	700, 700	700, 700
Generalization	0-12000	train	2×2	700, 280	10, 620
different flow 1	0-2000	test		700, 280	10, 620
	2000-4000	test		1000, 800	900, 700
	4000-6000	test		1400, 1000	400, 900
Generalization	0-12000	train	2×2	1000, 580	110, 920
different flow 2	0-2000	test		1000, 580	110, 920
	2000-4000	test		1000, 800	900, 700
	4000-6000	test		1400, 1000	400, 900
Generalization	0-12000	train	2×2	700, 700	700, 700
different flow 3	0-2000	test		700, 700	700, 700
	2000-4000	test		1000, 800	900, 700
	4000-6000	test		1400, 1000	400, 900
Train on 2×2	0-12000	train	2×2	700, 700	700, 700
test on 6×6	0-12000	test	6×6	700, 280, 260, 240, 780, 200	10, 620, 50, 660, 90, 700

4.1.2 Uniform flow on symmetric 2×2 network

The first road network is a symmetric environment with $N = 4$ traffic light agents defined by a 2×2 network Figure 4.3. Vehicle emissions from all route boundaries are equal and uniform throughout the entire training horizon. Due to symmetry in topology and vehicle emission, all agents' rewards are weighted equally and all agents receive the same (time-varying) distribution of observations. Hence all agents learn the same policy and/or value

functions. While this environment is more ideal than dynamic real-world traffic, it is a simple benchmark for investigating differences in algorithm performance.

4.1.3 Non-uniform traffic flow on 1×2 network

We implemented a second traffic control scenario with more challenging traffic dynamics than the previous one. One horizontal route and two vertical routes form two interactions where we installed two traffic lights Figure 4.2. We intentionally set different vehicle emissions for each route to create different incoming traffic densities for each traffic light. The horizontal route has 700 vehicles/hour, one vertical route on the left has only 10 vehicles/hour, while the second vertical route on the right has 600 per hour. This scenario reflects the common real-world setting where one main arterial road with dense traffic is adjacent and parallel to a smaller local road with sparse traffic. In this scenario, a good policy requires cooperative interaction between the two traffic lights. A greedy strategy for any individual will result in sub-optimal global reward, since the left traffic light will always prefer to let vehicles in the E-W direction pass through, which will lead to high incoming traffic for the second traffic light, which already has heavy incoming traffic in the vertical direction, resulting in high congestion. A better strategy for left traffic light is to consider the neighbor's situation and close the E-W route to some extent, to benefit the whole system at the cost of lower individual reward.

4.1.4 Generalization To Different Flows

Due to a variety of complex factors, such as time of day, location, and commuter schedule, traffic density is highly time-dependent. Since training on every possible traffic condition is not feasible, we investigated how well policies learned by each algorithm in one traffic condition generalize to different traffic conditions without further training. In the first experiment, we created three consecutive equal-duration time periods with different vehicle densities on the 2×2 road network, and directly deployed the model that

was trained under a static traffic flow in the same 2×2 network. In the test phase, the first period has the same traffic flow as the training phase. Denoting the flow as a vector $f := (\text{bot, up, left, right}) \in \mathbb{R}^4$ specifying the number of vehicles per hour on each vertical and horizontal route, traffic flow in the second and third periods are $f_{t_2} := (1000, 800, 900, 700)$ $f_{t_3} := (1400, 1000, 400, 900)$.

We further investigated the extent to which the generalization performance of a trained policy is affected by the specific traffic condition where it was trained. Specifically, we train QCOMBO with different flows in the same network topology (with one policy per training flow), and test the resulting policies under the same flow. We denote the i-th train-test combination ($i = 1, 2, 3$) as 4-component sequence $F^i = \{f^i_{t_0}, f^i_{t_1}, f_{t_2}, f_{t_3}\}$, where the first flow is the training flow followed by three test flows, and f_{t_2} and f_{t_3} are shared by all train-test combinations. Then the three train-test programs are: $F^1 = \{f^1_{t_0}, f^1_{t_1}, f_{t_2}, f_{t_3}\}$, $F^2 = \{f^2_{t_0}, f^2_{t_1}, f_{t_2}, f_{t_3}\}$, $F^3 = \{f^3_{t_0}, f^3_{t_1}, f_{t_2}, f_{t_3}\}$, where $f^1_{t_0} = f^1_{t_1} := (700, 280, 10, 620)$, $f^2_{t_0} = f^2_{t_1} := (1000, 580, 110, 920)$, $f^3_{t_0} = f^3_{t_1} := (700, 700, 700, 700)$ are training flows and the testing flow in the first $1/3$ period. Three programs use the same testing flow for the second and third periods (f_{t_2}, f_{t_3}), specified in Table 4.1. Time periods t_0, t_1, t_2, t_3 are 10000,1000,1000,1000 steps, respectively.

4.1.5 Policy generalization to a 6×6 network

Transportation networks of large cities feature many road intersections with dense and complex traffic flows. Directly training on simulations of such large real-world systems is not computationally practical due to the combinatorially-large state and joint-action spaces, regardless of whether independent or centralized training is used. Instead, we investigated the feasibility of training on a sub-network and directly transferring the learned policies without further training into the full environment. This approach was also considered by works in traditional traffic light simulation, whereby simple models were first formulated for regional traffic and then realigned to the whole system [33]. Rather than relying on

transfer planning [14], we tested the feasibility of directly deploying policies from subnetworks into the whole environment. To implement this experiment, we constructed a 6×6 traffic network with 36 traffic lights (Figure 4.4), and we injected the network with nonuniform traffic flows, which severely reduces the possibility that any traditional hand-designed traffic control plan can be the optimal policy. Policies trained via QCOMBO in the 2×2 network were directly tested on the 6×6 network containing 36 agents.

4.1.6 Algorithm implementations

We describe the implementation of all algorithms, using deep neural networks as flexible function approximators. To ensure that performance results can be attributed to algorithmic differences rather than to differences in neural network implementations, we ensure that all policy, value, and action-value functions have the same neural network architecture among all algorithms, to the extent allowed by each algorithm (e.g. QMIX requires a special hypernetwork architecture). Since the agent population is homogeneous, we employ parameter-sharing among all agents, who still act independently based on their individual observations [28].

The individual utility functions Q^n of IDQN, VDN, QMIX, and QCOMBO are represented by fully-connected three-layer neural networks, where the last layer has $|A|$ output nodes. QMIX has a two level hypernetwork that generates the weight matrix and bias vector from the global state s, to compute the inner product with each Q^n and produce one state action value $Q(s, \mathbf{a})$. VDN takes the sum of Q^n to calculate the total Q, which is minimized by the squared error loss. The critic of IAC is a value function V, which is used to estimate the TD error. The COMA uses a centralized Q minus a counterfactual baseline to compute the COMA policy gradient. The critics of COMA and IAC have the same three-layer neural network structure, similar to the Q-functions in IDQN, VDN, QMIX. Both IAC and COMA use the same actor network to approximate $\pi(o^n, a^n_{t-1})$, which is also a three-layer fully connected neural network that takes the agent's observation and last action, with a

softmax activation in the last layer. We give same information of agent's own observations, as well as the last actions, and one-hot vector of agent labels as input of value functions (Q^n of IDQN, QMIX, and VDN, V^n of IAC), we give Q of COMA the global state, all other agents' actions, and agent label as inputs. Similar to the use of RNNs for Q-functions in Sunehag, Lever, Gruslys, Czarnecki, Zambaldi, Jaderberg, Lanctot, Sonnerat, Leibo, and Tuyls [17] and Rashid, Samvelyan, Schroeder, Farquhar, Foerster, and Whiteson [18], and motivated by the possibility of periodic behavior of traffic, we also use RNN with GRU cells to approximate the Q^n of IDQN, VDN, QMIX and QCOMBO, and the $\pi(a_t|o_t)$ of IAC and COMA.

4.2 Results

We show learning curves for all algorithms and analyze differences in performance over learning. Previous work on deep RL for traffic signal control noted the potential for instability during training [14]. Hence it is important to show the stability of learning progress in addition to reporting final performance. Our learning curves were generated by conducting evaluation measurements periodically during training (i.e. exploit the current policy for 400 steps). Since it takes time to populate the road networks with vehicles, our learning curves start after 1000 simulation time steps. For every experiment, we ran a static policy (change light phase every 30s), and a random policy (keep or change the current phase every 5s) so that improvements due to learning can be clearly seen.

Table 4.2: Final Traffic Condition After Learning

	2×2 balanced						1×2 unbalanced					
	queue length		wait time		vehicle delay		queue length		wait time		vehicle delay	
QCOMBO	**1.80**	(0.05)	**0.03**	(0.00)	3.30	(0.01)	**1.12**	(0.96)	**0.14**	(0.20)	**2.08**	(0.11)
IDQN	23.92	(24.98)	191.98	(270.65)	3.44	(0.29)	29.78	(22.97)	44.63	(52.58)	2.37	(0.95)
IAC	41.89	(1.71)	9.16	(1.29)	3.45	(0.02)	34.37	(33.20)	42.13	(78.92)	2.27	(1.14)
COMA	2.10	(0.30)	0.05	(0.01)	**3.26**	(0.03)	7.72	(8.53)	0.84	(1.06)	2.19	(1.25)
QMIX	25.10	(32.10)	115.72	(163.57)	3.57	(0.31)	16.91	(19.11)	40.57	(58.34)	2.82	(0.12)
VDN	38.12	(29.13)	106.34	(105.83)	3.45	(0.30)	10.29	(15.08)	16.13	(34.09)	2.91	(0.47)
Random	26.57	(0.00)	2.49	(0.00)	3.65	(0.00)	25.15	(7.77)	4.96	(1.37)	2.81	(0.49)
Static	36.14	(0.00)	6.90	(0.00)	3.51	(0.00)	27.59	(4.51)	4.18	(1.60)	2.83	(0.02)

4.2.1 Static Traffic Flows

Figures 4.5 to 4.8 show the global reward in both the 2×2 and 1×2 road networks using both fully-connected and RNN neural networks. Over all flow and network configurations, QCOMBO attained the global optimal performance and is most stable among all algorithms. In general, policy-based methods COMA and IAC show lower variance and higher average reward than the value-based methods IDQN, VDN and QMIX. These Q-learning based methods may exhibit higher variance, since a small change of state-action-values can cause an action to be either selected or not by the deterministic policies. In contrast, stochastic policies learned by policy-based methods mitigate issues with state aliasing, and they are more robust as a small change of observation corresponds to a small change of the probability distribution over actions.

In the 2×2 environment Figure 4.5, QCOMBO and COMA converged to global optimal policies. Both two maintain good traffic condition throughout learning, as reflected by queue length, waiting time, and delay time of vehicles in Table 4.2. VDN performed better than the random and static policies, but worse than QCOMBO and COMA, likely because it can only learn a restricted set of linearly-combined Q functions. IDQN and IAC were not stable and failed to reach a global optimum, indicating that learning cooperation from the global reward is necessary even in this symmetric setting. QMIX diverged possibly due to the difficulty of stabilizing its hypernetwork. Surprisingly, all algorithms (with the exception of QMIX) improved with the use of RNN for policy or value functions (Figure 4.6), giving strong evidence that RNNs are especially suitable for handling history information and periodic waves of traffic that could occur in congested networks.

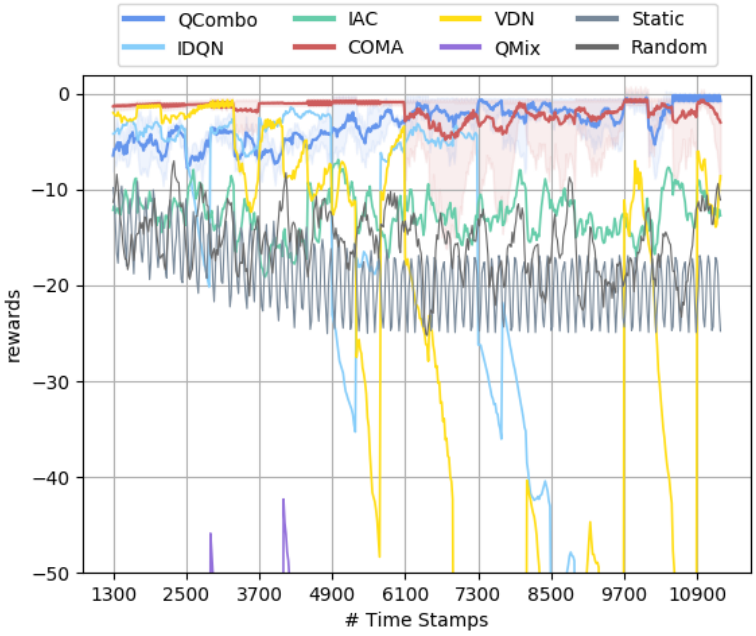


Figure 4.5: Global rewards: 2×2 network, 4 traffic lights

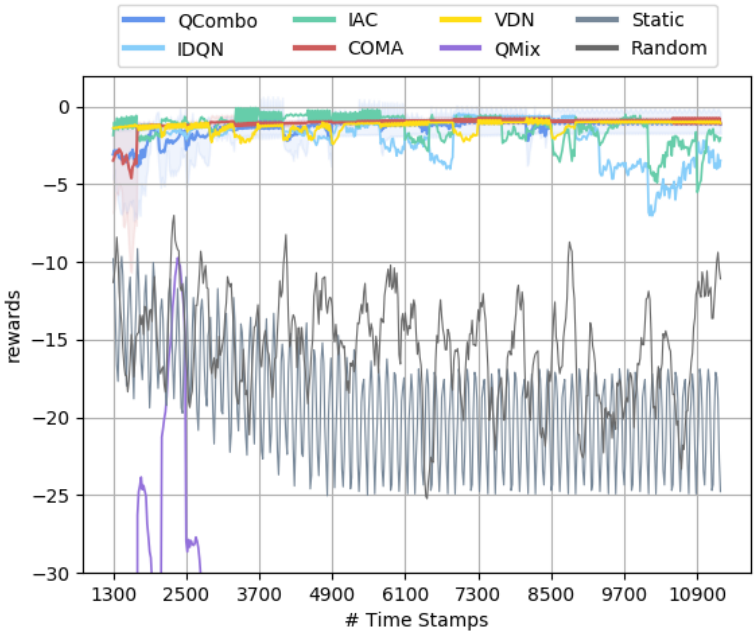


Figure 4.6: Global rewards: 2×2 using RNN

Performance differences between algorithms are more apparent in the 1×2 environment with non-uniform traffic flow (Figure 4.7). QCOMBO converged to the optimal policy at the end of training, exceeding the performance of all other algorithms. VDN began with high performance but struggled to maintain a good policy as more vehicles enter

the road. QCOMBO's higher performance and stability over IDQN shows that the new consistency regularization loss in QCOMBO helps to stabilize learning of independent utility functions by limiting their deviation from the centralized action-value function. The benefit of centralized learning (e.g. QCOMBO and COMA) over independent learning (e.g. IDQN, IDQN) is more apparent than in the 2×2 environment, since the non-uniform flow increases the impact of each agent on other agents' performance, resulting in the need for cooperation. IDQN, VDN and IAC exhibits oscillation, similar to behavior reported in [14]. Again, QMIX suffers from convergence issues when using a fully-connected network (RNN results shown in Figure 4.8).

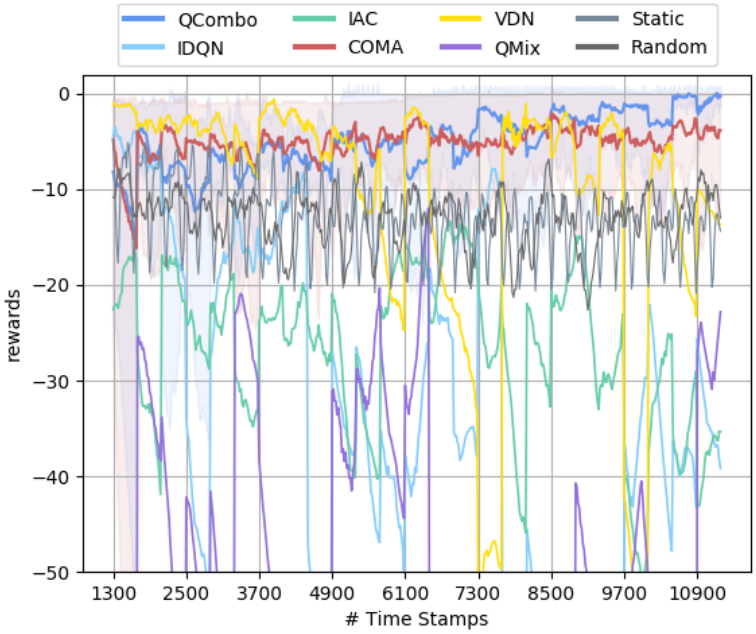


Figure 4.7: Global rewards: 2 traffic lights

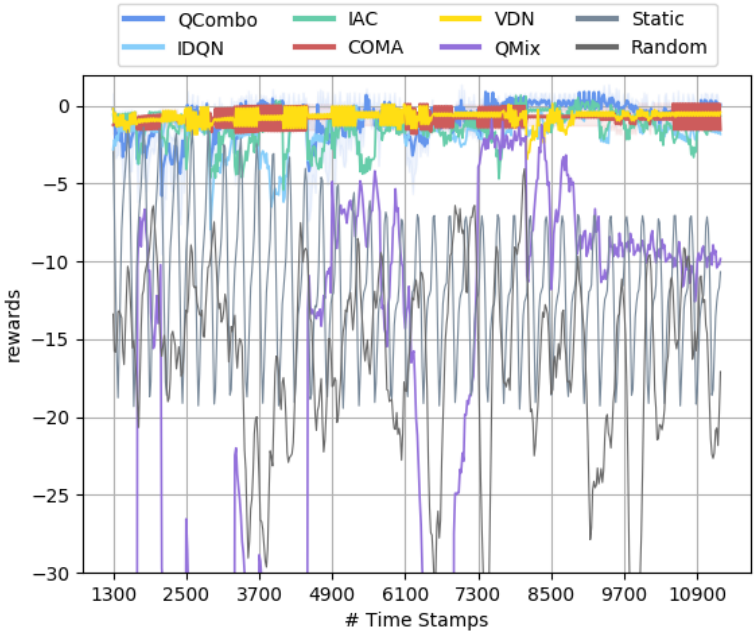


Figure 4.8: Global rewards: 2 traffic lights using RNN

Moreover, these differences in algorithm performance in 1×2 are manifest in significantly different actions (E-W or N-S phase) selected by the learned policies. In order to achieve cooperation, the left traffic light should open N-S and close E-W periodically to reduce the burden of incoming E-W traffic for the right light, who experiences heavy N-S traffic. Figures 4.9 and 4.10 show that QCOMBO and COMA achieve cooperation by turning off E-W traffic periodically (with low frequency, since it receives higher E-W than N-S traffic). However, IDQN greedily keeps E-W open for long durations, which is not the global optimal (Figure 4.11); IAC switches between the two phases almost equally (Figure 4.12); VDN switches to N-S too often than necessary (Figure 4.13); and QMix incorrectly chooses N-S more frequently than E-S (Figure 4.14).

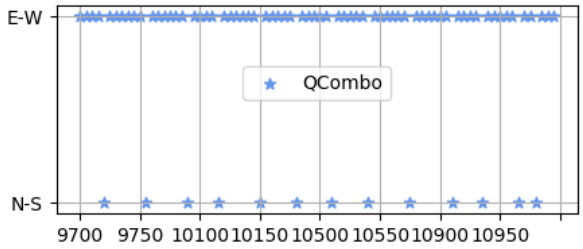


Figure 4.9: QCombo Policy

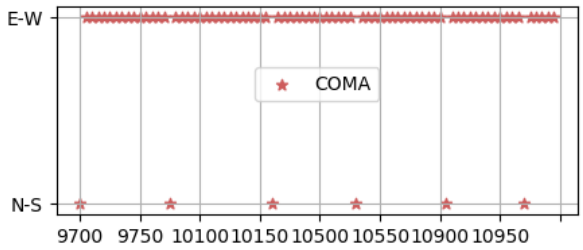


Figure 4.10: COMA Policy

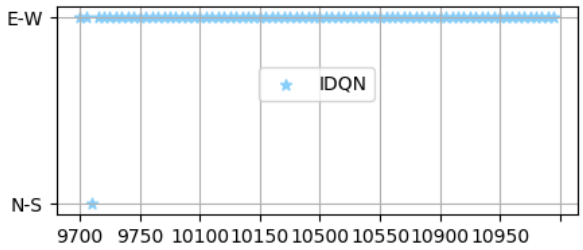


Figure 4.11: IDQN Policy

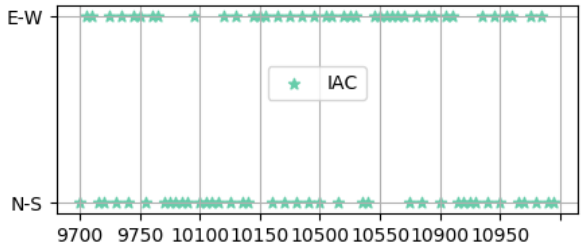


Figure 4.12: IAC Policy

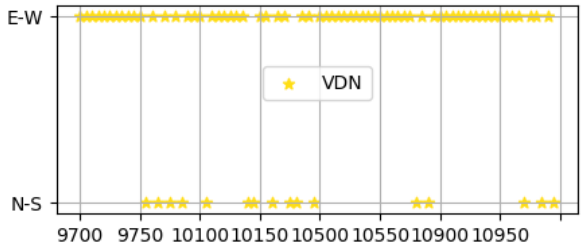


Figure 4.13: VDN Policy

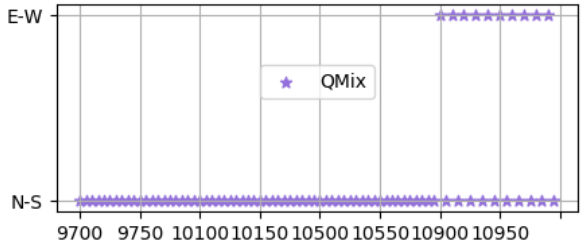


Figure 4.14: QMix Policy

4.2.2 Dynamic Traffic Flows

QCOMBO displayed the highest generalization performance, when trained on one traffic condition and deployed on two different test conditions Figure 4.15. As reflected by the decrease in performance of all policies when traffic flow changes at time step 2000, the test conditions were more difficult. While COMA and QCOMBO have equal performance on the training flow (first 1000 steps), QCOMBO generalized much more gracefully to the test conditions, possibly because the consistency regularization loss prevents overfitting to training conditions. VDN perform worse on the training flow than COMA but generalized better, despite experiencing a large drop in the second test flow when the vehicle density increases. IAC shows high variance on the test condition, while IDQN and QMIX could not adapt to new flows due to poor learning.

Figure 4.16 shows results of the second generalization experiment, where we evaluated QCOMBO policies trained on three different traffic flows, using the same test flow. For ease of reference, the first 1000 steps have the same flow as during training, while the new flows appear at $t = 2000$ and $t = 3000$. QCOMBO policies show flow invariance during the t_2 and t_3 period: trained on different traffic conditions, generalization performance on new unseen conditions exhibit only small variability. This gives evidence that QCOMBO is robust to generalization, such that performance on test conditions does not heavily depend on specific choices of training conditions.

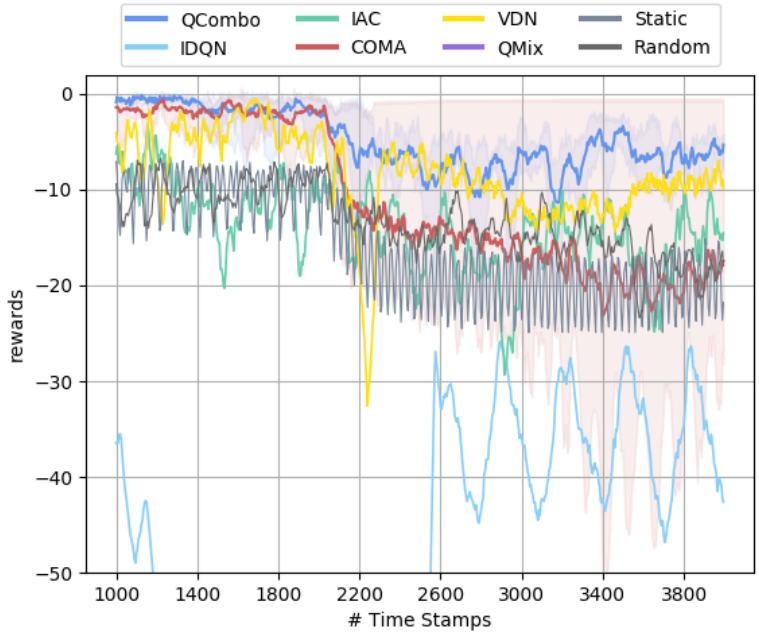


Figure 4.15: Generalization To Different Traffic Flows Among Algorithms

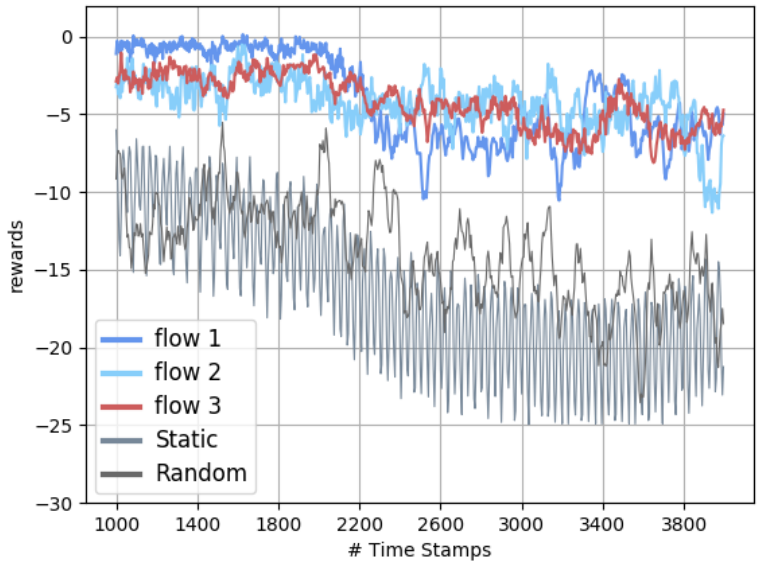


Figure 4.16: Impact of training conditions on performance in new test condition

4.2.3 Generalization To Larger Traffic Topology

We directly applied the QCOMBO policy trained in the 2×2 traffic network to the 6×6 network with 36 traffic light agents, which poses a significant generalization challenge due

to the decreased observability for any particular agent and the different traffic flow induced by the different network topology. Direct deployment in a larger systems is possible as QCOMBO learns decentralized policies. Figure 4.17 shows that QCOMBO's policy is able to maintain high and stable test performance, with almost negligible difference from its training performance. Surprisingly, it sometimes attains higher reward even than a policy that was trained specifically on the 6×6 environment. Hence, despite scalability issues with centralized training on many agents, we see that centralized training with few agents can still produce policies that generalize well to larger settings. This shows that QCOMBO's generalization to large system is not dependent on traffic flows, and gives strong evidence that a policy trained on a subset of a city road network can be deployed with little loss of performance at a larger scale.

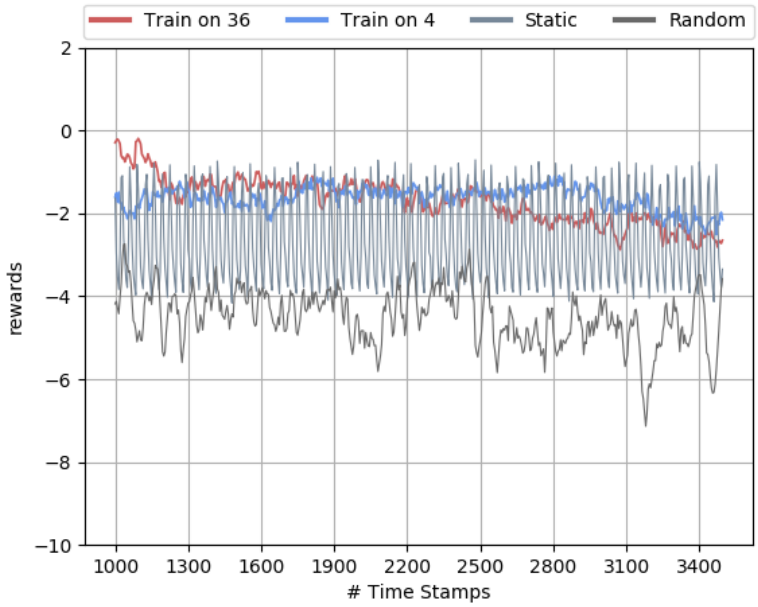


Figure 4.17: Policy trained in 2×2 network generalizes to 6×6 network

CHAPTER 5
CONCLUSION

In here, we proposed a novel MARL algorithm (QCOMBO) for traffic light control. QCOMBO combines the benefits of independent and centralized training, with a novel objective function that enforces consistency between a global action-value function and the weighted sum of individual optimal utility functions. We conducted detailed empirical evaluation of state-of-the-art MARL algorithms (IDQN, IAC, VDN, COMA, and QMIX) on network traffic light control under different map topologies and traffic flows, and showed that QCOMBO is a simple yet competitive approach to this real-world problem. Experiments also indicate that QCOMBO can be generalized with limited loss of performance to large traffic networks. Our work gives strong evidence for the feasibility of training cooperative policies for generalizable, scalable and intelligent traffic light control.

Appendices

APPENDIX A

ENVIRONMENT AND EXPERIMENT

A.1 SUMO Setup

One time step in SUMO corresponds to 0.1s in the real world. We limit the traffic light to control vehicles going straight only. Table A.1 has the configuration of SUMO environment.

Table A.1: The SUMO Configuration

	Parameter	Value	Description
vehicle	maxSpeed	35m/s	The maxium alllowed speed in the lane
	minGap	2m	The minimum empty space that vehicles are allowed to have
	tau	1s	The minimum time gap of tau between the rear bumper of vehicle's leader and its own front-bumper + minGap to assure the possibility to brake in time when its leader starts braking and it needs tau seconds reaction time to start breaking as well.
	spacing	uniform	The positioning of vehicles in the network relative to one another
	accel	$1.0m/s^2$	The acceleration ability
	decel	$1.5m/s^2$	The deceleration ability
	speed_factor	1	The vehicles expected multiplicator for lane speed limits
	speed_dev	0.1	The deviation of the speedFactor
	car_follow_model	IDM	The numerical integration method used to control the dynamic update of the simulation, the intelligent driver model results in very conservative lane changing gap acceptance.
traffic light	phase	green, yellow, red	The traffic light alternating phase
	green	30s	The green light duration
	yellow	3s	The yellow light duration
	red	30s	The red light duration
	tls_type	static	Static traffic lights are traffic lights with pre-defined phases, they cannot dynamically adjust according to traffic needs; they simply follow the same pattern repeatedly.

A.2 Reward Definition

The individual reward $R^n = c_1 \times ql^n + c_2 \times wtl^n + c_3 \times dl^n + c_4 \times eml^n + c_5 \times fl^n + c_6 \times vl^n$ is a weighted linear combination of features $ql^n, wtl^n, dl^n, eml^n, fl^n, vl^n$ with weights $c_1, c_2, c_3, c_4, c_5, c_6$, given for each Δ_t=5s interval.

- $ql^n := \sum_{\Delta_t} \sum_l q_l^n$ is the sum of queue length of all incoming lanes, weighted by $c_1 = -0.5$

- $wtl^n := \sum_{\Delta_t} \sum_l wt_l^n$ is the sum of vehicle waiting time among all incoming lanes, with $c_2 = -0.5$.

- $dl^n := \sum_{\Delta_t} \sum_l delay_l^n$ is the sum of vehicle delay of all incoming lanes queue, with $c_3 = -0.5$.

- eml^n, the number of vehicles that have emergence stops on the traffic light lanes over Δ_t, a vehicle is counted in this category if its current speed more than 4.5m/s less than the 1 second before, and $c_4 = -0.25$

- fl^n, the number of times that traffic lights change its phase over Δ_t, and $c_5 = -1$.

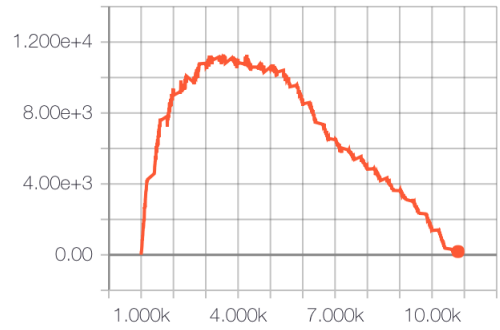
- vl^n, the number of vehicles that pass the traffic light over Δ_t, and $c_6 = 1$.

APPENDIX B

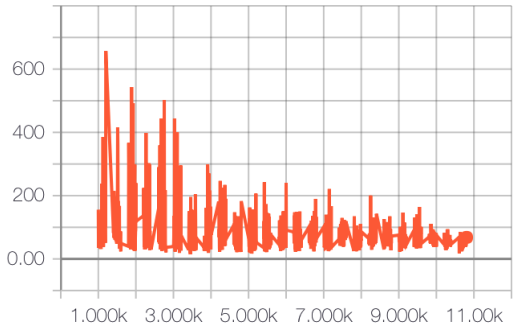
QCOMBO

B.1 QCOMBO Loss Curve

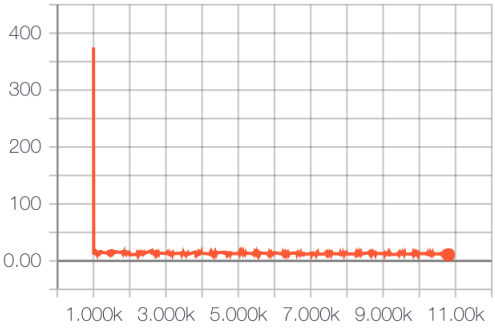
Figure B.1 shows the learning curve for loss in QCOMBO.



(a)



(b)



(c)

Figure B.1: Training Loss on 1×2 map: (a) consistency loss; (b) global loss; (c) individual loss;

APPENDIX C
ARCHITECTURE AND TRAINING

C.1 Fully Connected Neural Network

For standardization, the Q^n of IDQN, VDN, QMIX, QCOMBO and V^n of IAC use the same nerual network architecture, which is a three-layer fully-connected neural network, with 256 units in the hidden layers and ReLu activation. The output has $|A|$ units to represent the agent's state action value function output. The input is the agent's own observation vector o^n, its last action a^n_{t-1}, and one hot agent label n. The actors of IAC and COMA use the same neural network structure, which is a three-layer fully-connected network with 64 hidden units and ReLu activation. The output of actor neural network uses the softmax activation, and the input is o^n. All agents share the same parameter θ. The global $Q^{\boldsymbol{\pi}}$ of QCOMBO is approximated by a three-layer fully-connected neural network with 256 hidden units, its input is the global state s and all agents' action $\mathbf{a}$, and the output is a single node. The $Q^{\boldsymbol{\pi}}$ of COMA also has three layers, and has inputs $\{s, \mathbf{a}^{-n}, n, o^n\}$. The learning rate is 0.001 for Q and V, and 0.0001 for the actor network. The target network is updated gradually in every training step as a sum of 1% of online network parameters and 99% of current target network parameters. All the network training uses the Adam optimizer.

C.2 RNN Network Structure

An alternative implementation of the algorithms use RNN with GRU cell to approximate the Q^n of IDQN, QMIX, VDN, QCOMBO, and the actors of COMA and IAC. The RNN has 64 hidden units with a Relu activation function, the input of RNN is linearly trans-

formed from input space to 64 units, and the output of the hidden layer is linearly transformed into $|A|$ outputs to represent each discrete action. The hidden state is retained for each time step and also re-used for next training batch.

C.3 Training Strategy

All the algorithms have same training strategy. The total time horizon for training and online evaluation is limited to 12K SUMO time steps. We limit the agents to make decisions for every 5 times steps to avoid high-frequency flickering of traffic lights. For each training run, the environment runs freely for 1000 steps using a random policy to populate the road network with vehicles before training begins. Then we alternate between training and evaluation, with each training and evaluation period lasting 400 time steps. During every training step (once per 5 time steps), the algorithm updates parameters of the the current networks via stochastic gradient descent with 100 minibatches of 30 samples from the replay buffer. The replay buffer is a first-in first-out queue containing the 1000 most recent samples. If the RNN framework is involved, we use a different sampling strategy: 1. every training cycle contains 6 consecutive periods; 2. in each period, we collect 30 consecutive samples and feed them into the RNN in the order of their corresponding time periods from the earliest to the latest; 3. after we finish one period parameter learning with 30 time steps data, we record the hidden state and use it to initialize the hidden state for the next learning step. The current action is selected through an ϵ-greedy behavior policy with ϵ starting with 0.9 and a decay geometric factor of 0.995. ϵ is decayed every training cycle and fixed within a cycle. A training cycle is followed with an on-policy evaluation period. During the evaluation period of 400 time steps, the system executes the current policy and stops decaying the ϵ. We record rewards only during the last 200 of the 400 time steps, to measure the steady-state of the system under the current policy.

REFERENCES

[1] F. Guerrini, "Traffic congestion costs americans $124 billion a year, report says," *Forbes, October*, vol. 14, 2014.

[2] L. McNew, *Americans will waste $2.8 trillion on traffic by 2030 if gridlock persists*, 2014.

[3] I. Porche and S. Lafortune, "Adaptive look-ahead optimization of traffic signals," *Journal of Intelligent Transportation System*, vol. 4, no. 3-4, pp. 209–254, 1999.

[4] C. Gershenson, "Self-organizing traffic lights," *arXiv preprint nlin/0411066*, 2004.

[5] S.-B. Cools, C. Gershenson, and B. DHooghe, "Self-organizing traffic lights: A realistic simulation," in *Advances in applied self-organizing systems*, Springer, 2013, pp. 45–55.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[8] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[9] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142*, 2016.

[10] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[11] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2018, pp. 2496–2505.

[12] M. LIU, J. DENG, M. XU, X. ZHANG, and W. WANG, "Cooperative deep reinforcement learning for tra ic signal control," 2017.

[13] I. Arel, C. Liu, T Urbanik, and A. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intelligent Transport Systems*, vol. 4, no. 2, pp. 128–135, 2010.

[14] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, 2016.

[15] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[16] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[17] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[18] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 4295–4304.

[19] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," *IET Intelligent Transport Systems*, vol. 11, no. 7, pp. 417–423, 2017.

[20] X. Liang, X. Du, G. Wang, and Z. Han, "Deep reinforcement learning for traffic light control in vehicular networks," *arXiv preprint arXiv:1803.11115*, 2018.

[21] P. Balaji, X German, and D. Srinivasan, "Urban traffic signal control using reinforcement learning agents," *IET Intelligent Transport Systems*, vol. 4, no. 3, pp. 177–188, 2010.

[22] Y. K. Chin, N. Bolong, A. Kiring, S. S. Yang, and K. T. K. Teo, "Q-learning based traffic optimization in management of signal timing plan," *International Journal of Simulation, Systems, Science & Technology*, vol. 12, no. 3, pp. 29–35, 2011.

[23] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," Stanford InfoLab, Tech. Rep., 1999.

[24] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Method-

ology and large-scale application on downtown toronto," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1140–1150, 2013.

[25] Y. Liu, L. Liu, and W.-P. Chen, "Intelligent traffic light control using distributed multi-agent q learning," *arXiv preprint arXiv:1711.10941*, 2017.

[26] N. Geroliminis, C. F. Daganzo, *et al.*, "Macroscopic modeling of traffic in cities," in *86th Annual Meeting of the Transportation Research Board, Washington, DC*, 2007.

[27] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, e0172395, 2017.

[28] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *arXiv preprint arXiv:1705.08926*, 2017.

[29] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[30] J. Yang, A. Nakhaei, D. Isele, H. Zha, and K. Fujimura, "Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning," *arXiv preprint arXiv:1809.05188*, 2018.

[31] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: Architecture and benchmarking for reinforcement learning in traffic control," *arXiv preprint arXiv:1710.05465*, 2017.

[32] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

[33] K. N. J. Esser and M Rickert, *Large-scale traffic simulations for transportation planning*. World Scientific, Singapore, 2000.